Certification of Adaptive Flight Control Software

Michael Richard, SMART-T Chief Engineer
NASA Dryden Flight Research Center

michael.richard@nasa.gov

V. Santhanam, Technical Fellow
Peggy Wright, DAS Software AR
Boeing Integrated Defense Systems - Wichita

Explore. Discover. Understand.

Intelligent Flight Control Systems

- Goals of IFCS Project
  - Demonstrate revolutionary concepts that can efficiently improve aircraft stability and control in both normal and failure conditions
  - Advance adaptive flight control technology for future aerospace systems designs
  - Determine obstacles to FAA certification of adaptive software
- Collaborative effort among:
  - NASA Dryden Flight Research Center
  - NASA Ames Research Center
  - Boeing Phantom Works in St. Louis
  - Institute for Scientific Research (ISR)
  - West Virginia University
  - Georgia Institute of Technology

Verification and Validation

- SMART-T (Strategic Methodologies for Autonomous & Robust Technology Testing) project working with IFCS
  - Researching and developing V&V tools and guidelines for adaptive control systems
- Partners Include:
  - NASA Dryden Flight Research Center
  - NASA Ames Research Center
  - Boeing Phantom Works in St. Louis
  - Boeing Integrated Defense Systems - Wichita
  - Institute for Scientific Research (ISR)
  - Case Western Reserve University

FAA Certification

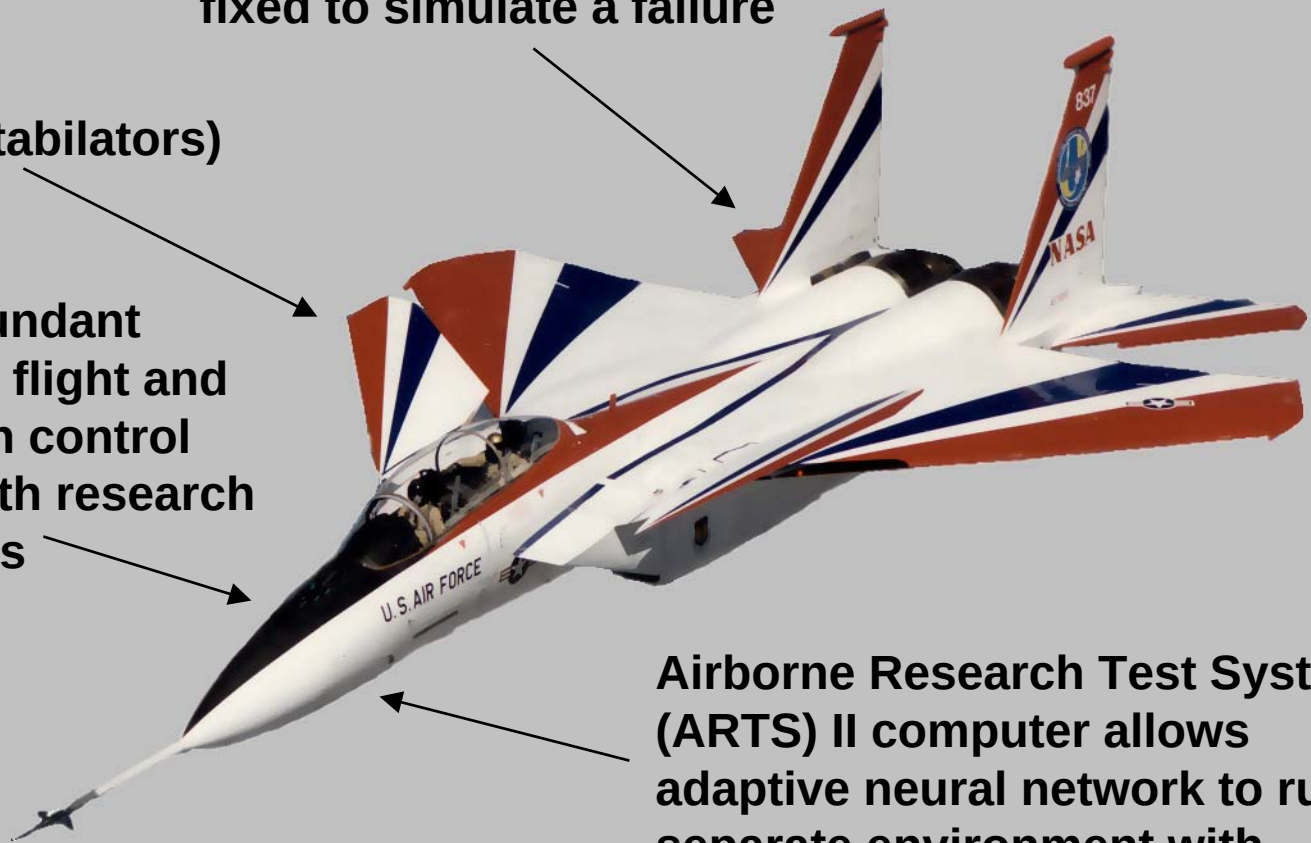
- NASA and Boeing have undertaken a study to compare some of the artifacts and software created for the IFCS project against the certification guidelines in DO-178B
- We are interested in your opinion and welcome questions or concerns
- FAA participation is needed so that research can continue in the proper direction

NASA 837 NF-15B

Right stabilator can be fixed to simulate a failure

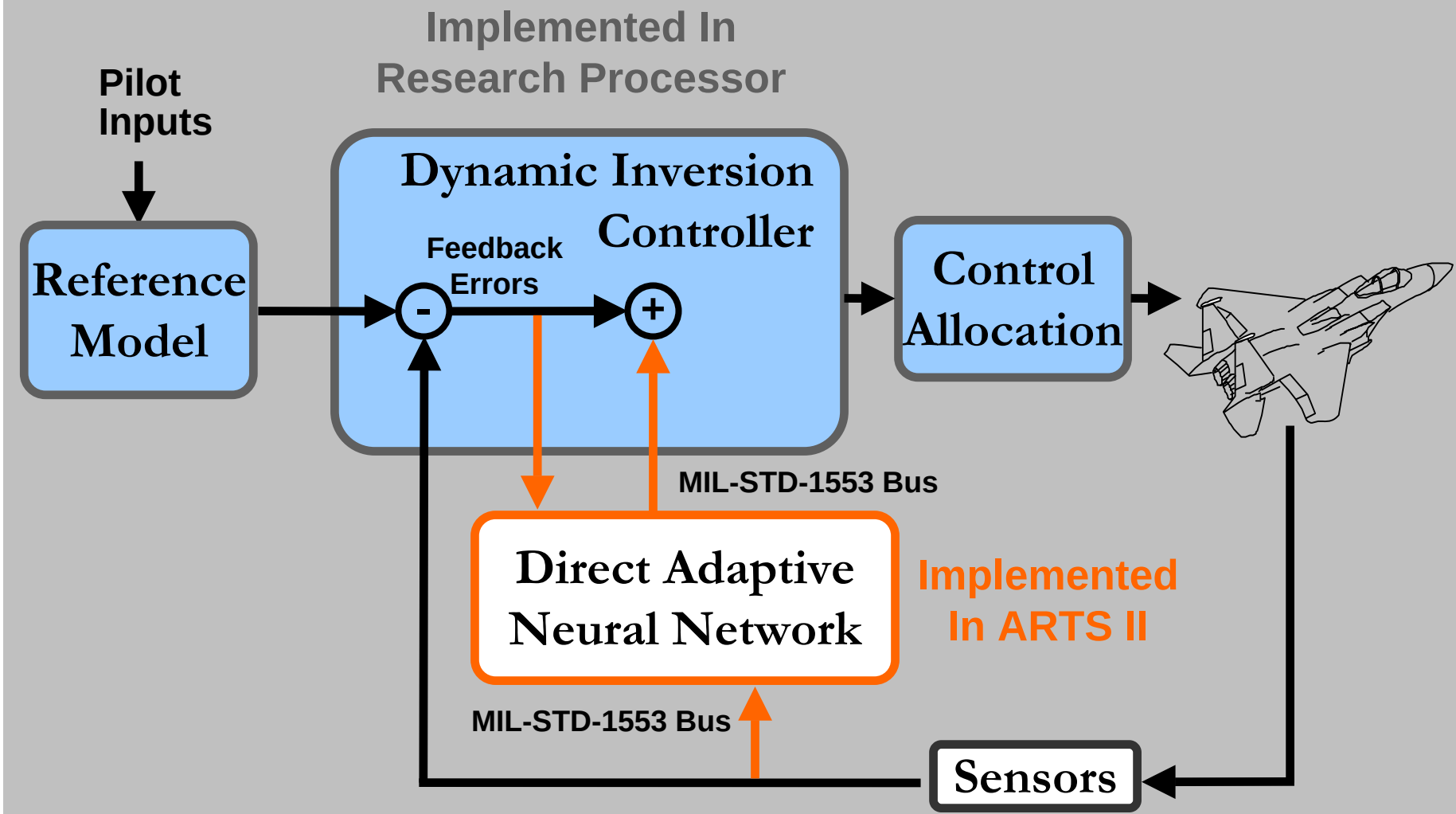
Canards (F/A-18 stabilators)

Quad-redundant integrated flight and propulsion control system with research processors



Airborne Research Test System (ARTS) II computer allows adaptive neural network to run in separate environment with limited authority

Adaptive Flight Control Diagram



Adaptation

- Neural networks are universal approximators

  Output = Σ [**Weights** $*$ **Basis Functions (Inputs)**]

  – Weights are determined by an adaptation or learning rule
- The IFCS neural network directly augments the controller commands to reduce feedback errors
  – Single-layer feedforward linear neural network that adapts online
  – It needs a reference model to calculate feedback error

    Error = Expected Performance – Sensed Performance
  – Neural network weights continuously adapt to minimize the errors

    Change in Weights = $Gain_1 * (Error +_i Error) * Inputs$

    $+ Gain_2 * abs(Error +_i Error) * Weights$
  – Adaptation stops when the errors are small
  – No aero parameter estimation or failure identification is needed

Flight Test Plan

- Limited flight envelope
- Assess handling qualities of controller without adaptation
- Activate adaptation and assess changes in handling qualities
- Demonstrate the ability of the system to adapt to failures
  - Jammed control surface (fixing right stabilator)
  - Changes in aircraft aerodynamics and stability (modifying lift from canards)
- Report on “real world” experience with a neural network based flight control system

Can Adaptive Flight Control Software be Certified to DO-178B Level A?

Vdot Santhanam

Technical Fellow

Boeing Integrated Defense Systems-Wichita

July 2005

Adaptive vs. Conventional Software

- What makes Adaptive Flight Control software different from conventional software?
  - Conventional software starts in the same exact state on each power up
  - Adaptive software can have its initial conditions vary over its operational lifetime as a result of “learning”
- This basic difference has led to some common myths about adaptive software…

Adaptive Software: Myth #1

- Adaptive software is “self-modifying” software
  - This is a poor way of characterizing the fact that adaptive software could have different start-up states over time
  - The term “self-modifying code” describes software which changes its own instructions before executing them, making it a challenge to test such software
  - ❖ Most adaptive algorithms store their knowledge in conventional data structures, not in their object code

Adaptive Software: Myth #2

- Adaptive software is non-deterministic
  - Perhaps a (poor) characterization of the learning attribute
  - Or, perhaps a reflection of the fact that adaptive software is often aimed at handling unforeseen configurations
  - ❖Either way, the statement is incorrect
    - ❖Flight control software, adaptive or not, must deal with situations that it was not verified to deal with
    - ❖Adaptive software, starting with the same initial condition and given the same set of inputs, will behave exactly the same way each time

Adaptive Software: Myth #3

- Adaptive software can grow unboundedly
  - This claim stems perhaps from the fact that some of the adaptive software continues to learn forever
  - ❖Not all such learning algorithms use ever-growing data structures
    - – The learning algorithm that we studied uses a fixed set of weights which are continually *refined* during learning
  - ❖Even the algorithms that use a growing set of nodes need not grow forever
    - – Many can be restricted to grow within preset limits and still retain the learning and stability characteristics

Adaptive Software: Myth #4

- Adaptive software is impossible to test
  - Any software with a large number of internal states poses a challenge to the testers
    - States must be divided into a smaller number of equivalence classes
    - Inputs must be chosen to exercise the software in each equivalence class
  - ❖ It is no more difficult to test adaptive software than it is to test conventional software with a comparable number of states

Adaptive Software: A Case Study

- We studied the verifiability of a portion of an adaptive flight control system
  - The system uses a single-layer online adaptive feedforward linear neural network
  - The system is modeled using MATRIXx
  - The flight code is auto-generated using MATRIXx's autocoder
- Our objective was to determine if there are inherent difficulties in meeting DO-178B requirements for that software

Case Study Details

- It was quickly determined that the autocode generated had many shortcomings
  - Autocode used pointers lavishly, making it difficult to test the code
  - Traceability of autocode to model blocks was obscured by unnecessary complexity
  - These shortcomings were not specific to neural network being modeled, but they got in the way of verification
- Developed a prototype autocoder to generate clean, traceable code

Case Study Results (preliminary)

- The case study has revealed interesting facts
  - The portion of code that corresponded to the neural network algorithm was indistinguishable from the rest
  - Generating test cases to achieve structural coverage was no more (nor less) difficult because the software was adaptive
  - Traceability of source code to the higher level model was just as easy for adaptive algorithms as it was for conventional portions of the model

MC/DC and Adaptive Software

- It is generally believed that achieving Modified Condition/Decision Coverage is the greatest challenge for Level A software
  - We found that it would be no more difficult to achieve this than it would be for any other software of comparable complexity

The *Real* Challenges

- What then are the real challenges we see in certifying adaptive software?
  - Showing that the neural networks do meet the high level requirements for flight control system
    - Validation, Traceability
  - Stability of the learning algorithms
    - Acceptable behavior within the operational envelope
  - Dividing the state space into a reasonable number of equivalence classes from which to draw test cases
    - Beyond structural coverage, how much testing is enough testing?

❖Getting regulatory authorities to believe all these are possible

Software Assurance for Adaptive Neural Networks in Aerospace Applications

Peggy Wright
Designated Alteration Station (DAS)
Software Authorized Representative (AR)
Boeing Integrated Defense Systems - Wichita

Overview

- Adaptive Neural Network (ANN) Attributes
- ANN Software Study
- Obstacles to FAA Acceptance
- Why Should We Pursue ANN?
- Proposal for ANN Software Assurance

Adaptive Neural Networks (ANN) Flight Control Software Background

- ANN software began to proliferate in the 1980s
- Is flying now on experimental aircraft
  - F-15, experimental aerial vehicles
- Adapts to the specific aircraft and its physical conditions to facilitate flight stability and improve command tracking
- This software has not yet been accepted on any FAA certification project – is it safe?

ANN Flight Control Software Why Is It Needed?

- Provides quick response to improve tracking performance while maintaining aircraft stability during:
  - Battle damage
  - System failure
  - Degradation over time
  - High-risk situations such as Space Missions
- Users say it provides a viable emergency strategy
  - Greater flexibility
  - Rapid response
  - Adaptability to the aircraft and situation
  - It works!

F-15 Adaptive Neural Network Software Study

- In order to address the question of safety:
  - Boeing Phantom Works Team invited the Wichita Software and Languages Technology group to study their software to evaluate verifiability
  - We performed a Gap Analysis
  - We studied the software source code
  - We studied the verification methods

F-15 ANN Flight Control Software Study Results

- Boeing has found that this software is deterministic
  - Computes results in bounded time that are the inevitable consequence of the inputs
  - Follows its algorithms in regulated ways like any software
  - Does not change its own code
  - Is different only because of stored calculated values.
- It can be shown to perform correctly within its bounds
- Tests are repeatable
  - Learns behavior by storing the values it has calculated
  - The stored values must be considered part of the test inputs
- It has a monitor to limit the bounds of its computed data to assure safety

F-15 ANN Flight Control Software Study Results (cont.)

- Gap Analysis was performed to evaluate software development processes against DO-178B objectives
- Gaps were found..
    - Plans for QA do not specify all of the activities expected at level A.
    - Although some project plans clearly state the needed verification and QA activities will be performed, they also limit their scope to only a part of the data, due to limitations in authority among multiple corporate/government entities
    - Items like Additional Considerations (DO-178B Section 12) are not addressed.
  - Overall, more rigorous processes are needed
- ..BUT no insurmountable problems were discovered.

Obstacles to ANN Acceptance

- Difficult to Show Compliance in the General Case
  - Neural Nets have a “Bad Rep”
    - Have been called Non-Deterministic, Self-Modifying, and Unverifiable
  - Knowledge is learned and remembered over time
  - The software behavior changes as it adapts to the aircraft
    - Behavior can change over time and across aircraft
  - There is no widely accepted methodology for verification
    - Software does not behave the same every time, depending on the learned state of the neural net
    - Complexity
  - There is no FAA guidance for software assurance

No Obstacles Found for F-15 ANN Software

- Complexity
  - Not unlike other flight control software
  - Is comparable to a Kalman Filter
- Verifiability
  - Tests are repeatable when stored data is considered part of test input
- Self-Modifying
  - Code is not changed – only computed data values change.
- Non-Deterministic?
  - The Boeing study shows this is false!

Why Should We Pursue ANN?

- ANN Presents a Unique Software Opportunity
  - Has been shown to be effective in F-15 and other programs
  - Is deterministic and verifiable (including MC/DC) in the F-15 application
  - Offers flexible new capabilities that enhance safety of flight

Will ANN Be Certified ?

- The time has come!
  - Experience with technology tells us that it is inevitable that Adaptive Neural Network Software WILL BE on future aircraft systems.
    - For Improved Safety of Flight
  - Software safety is the responsibility of this community
  - A process to assure safety of this powerful software is needed

Proposal

- We propose that NASA, DoD, and Industry join with the FAA to develop ANN best practices guidance material
  - (Such as was done recently for Object-Oriented technology)

- We believe this is the best way to achieve:
  - Software assurance for ANN software
  - Safe technology transition for this powerful paradigm